

Problem 1

```
#include "Polynomial.h"
#include <math.h>

double Polynomial::operator()(double aX) const {
    double lResult = 0;

    for (int i = fDegree; i >= 0; i--)
    {
        lResult += fCoeffs[i] * pow(aX, i);
    }

    return lResult;
}

Polynomial Polynomial::getDifferential() const {
    Polynomial lResult;

    for (int i = fDegree; i >= 0; i--)
    {
        lResult.fCoeffs[i - 1] = fCoeffs[i] * i;
    }

    if (fDegree - 1 >= 0)
    {
        lResult.fDegree = fDegree - 1 <= MAX_POLYNOMIAL ? fDegree - 1 : MAX_POLYNOMIAL;
    }
    else
    {
        lResult.fDegree = 0;
    }

    return lResult;
}

Polynomial Polynomial::getIndefiniteIntegral() const {
    Polynomial lResult;

    if (fDegree + 1 <= MAX_POLYNOMIAL)
    {
        lResult.fDegree = fDegree + 1;
    }
    else
    {
        lResult.fDegree = MAX_POLYNOMIAL;
    }

    for (int i = lResult.fDegree - 1; i >= 0; i--)
    {
        lResult.fCoeffs[i + 1] = fCoeffs[i] / ((double)i + 1);
    }

    return lResult;
}
```

```

double Polynomial::getDefiniteIntegral(double aXLow, double aXHigh) const {
    double lLow = 0;
    double lHigh = 0;

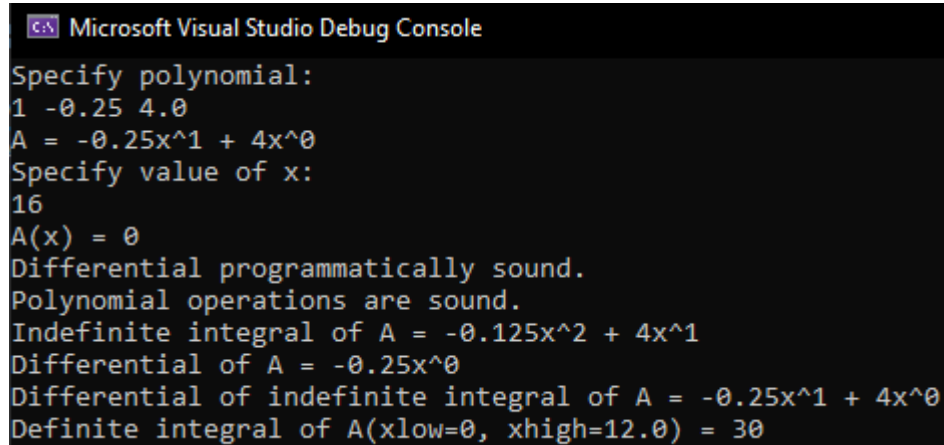
    for (int i = fDegree; i >= 0; i--)
    {
        lLow += fCoeffs[i] / ((double) i + 1) * pow(aXLow, i + 1);
    }

    for (int i = fDegree; i >= 0; i--)
    {
        lHigh += fCoeffs[i] / ((double)i + 1) * pow(aXHigh, i + 1);
    }

    return lHigh - lLow;
}

```

Output



```

C:\N Microsoft Visual Studio Debug Console
Specify polynomial:
1 -0.25 4.0
A = -0.25x^1 + 4x^0
Specify value of x:
16
A(x) = 0
Differential programmatically sound.
Polynomial operations are sound.
Indefinite integral of A = -0.125x^2 + 4x^1
Differential of A = -0.25x^0
Differential of indefinite integral of A = -0.25x^1 + 4x^0
Definite integral of A(xlow=0, xhigh=12.0) = 30

```

Problem 2

```
#include "Combination.h"

Combination::Combination(unsigned int aN, unsigned int aK) {
    fN = aN;
    fK = aK;
}

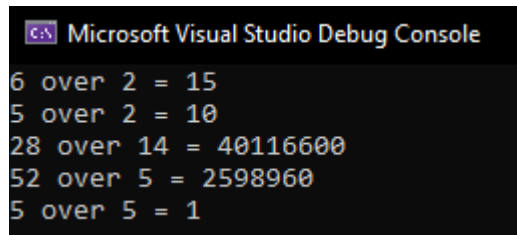
unsigned int Combination::getN() const {
    return fN;
}

unsigned int Combination::getK() const {
    return fK;
}

unsigned long long Combination::operator()() const {
    if (fK > fN) return 0;
    if (fK == fN) return 1;

    unsigned long long lResult = 1;
    for (int k = 1; k <= fK; k++)
    {
        lResult *= ((unsigned long long)fN - k + 1);
        lResult /= k;
    }
    return lResult;
}
```

Output



C:\> Microsoft Visual Studio Debug Console

```
6 over 2 = 15
5 over 2 = 10
28 over 14 = 40116600
52 over 5 = 2598960
5 over 5 = 1
```

Problem 3

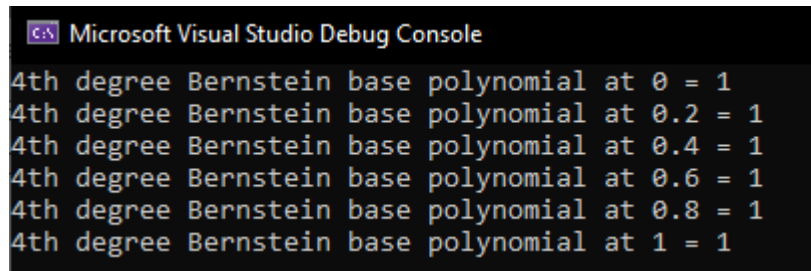
```
#include "BernsteinBasePolynomial.h"
#include <math.h>

BernsteinBasePolynomial::BernsteinBasePolynomial() : fFactor(0, 0) {}

BernsteinBasePolynomial::BernsteinBasePolynomial(unsigned int aV, unsigned int aN) : fFactor(aN,
aV) {}

double BernsteinBasePolynomial::operator()(double aX) const {
    return fFactor() * pow(aX, fFactor.getK()) * pow((1 - aX), (fFactor.getN() -
fFactor.getK()));
}
```

Output



The screenshot shows the Microsoft Visual Studio Debug Console with the following output:

```
C:\> Microsoft Visual Studio Debug Console
4th degree Bernstein base polynomial at 0 = 1
4th degree Bernstein base polynomial at 0.2 = 1
4th degree Bernstein base polynomial at 0.4 = 1
4th degree Bernstein base polynomial at 0.6 = 1
4th degree Bernstein base polynomial at 0.8 = 1
4th degree Bernstein base polynomial at 1 = 1
```